Analysis of tensor approximation for compression-domain volume visualization

Ballester-Ripoll, Rafael ; Suter, Susanne K ; Pajarola, Renato

Abstract: As modern high-resolution imaging devices allow to acquire increasingly large and complex volume data sets, their effective and compact representation for visualization becomes a challenging task. The Tucker decomposition has already confirmed higher-order tensor approximation (TA) as a viable technique for compressed volume representation; however, alternative decomposition approaches exist. In this work, we review the main TA models proposed in the literature on multiway data analysis and study their application in a visualization context, where reconstruction performance is emphasized along with reduced data representation costs. Progressive and selective detail reconstruction is a main goal for such representations and can efficiently be achieved by truncating an existing decomposition. To this end, we explore alternative incremental variations of the CANDECOMP/PARAFAC and Tucker models. We give theoretical time and space complexity estimates for every discussed approach and variant. Additionally, their empirical decomposition and reconstruction times and approximation quality are tested in both C++ and MATLAB implementations. Several scanned real-life exemplar volumes are used varying data sizes, initialization methods, degree of compression and truncation. As a result of this, we demonstrate the superiority of the Tucker model for most visualization purposes, while canonical-based models offer benefits only in limited situations.

DOI: https://doi.org/10.1016/j.cag.2014.10.002

Analysis of Tensor Approximation for Compression-Domain Volume Visualization

Rafael Ballester-Ripoll, Susanne K. Suter, Renato Pajarola

Visualization and MultiMedia Lab, Department of Informatics, University of Zürich

Abstract

As modern high-resolution imaging devices allow to acquire increasingly large and complex volume data sets, their effective and compact representation for visualization becomes a challenging task. The Tucker decomposition has already confirmed higher-order tensor approximation (TA) as a viable technique for compressed volume representation; however, alternative decomposition approaches exist. In this work, we review the main TA models proposed in the literature on multiway data analysis and study their application in a visualization context, where reconstruction performance is emphasized along with reduced data representation costs. Progressive and selective detail reconstruction is a main goal for such representations and can efficiently be achieved by truncating an existing decomposition. To this end, we explore alternative incremental variations of the CANDECOMP/PARAFAC and Tucker models. We give theoretical time and space complexity estimates for every discussed approach and variant. Additionally, their empirical decomposition and reconstruction times and approximation quality are tested in both C++ and MATLAB implementations. Several scanned real-life exemplar volumes are used varying data sizes, initialization methods, degree of compression and truncation. As a result of this, we demonstrate the superiority of the Tucker model for most visualization purposes, while canonical-based models offer benefits only in limited situations.

Keywords: Tensor approximation, volume visualization, higher-order decompositions, canonical decomposition, Tucker decomposition, tensor rank truncation

1. Introduction

Volume visualization has become an integral part of many research disciplines as a tool to visually analyze, explore and inspect large 3D data sets. Advanced data acquisition devices, however, continue to produce data sets exceeding the capacity of standard volume rendering techniques and graphics (GPU) memory. Hence, it is critical to develop and evaluate suitable adaptive 3D volume data reduction techniques.

Data reduction is often achieved by using compact data representation models. A typical approach is to transform the input volume data by means of a mathematical framework into a compact data representation using fewer data coefficients, optionally followed by bit-level compression. To visualize the data, the inverse transformation is applied to reconstruct the volume to an approximation meeting user-defined quality requirements. This decomposition-reconstruction process is usually highly asymmetric. That is, the data decomposition step is an offline process (not time critical), while the reconstruction process has to be performed online for real-time rendering.

Compact data models typically decompose the input data into a different domain that is more suitable for data reduction approaches like thresholding, quantization and encoding. Such decompositions capture the energy of the input data (signal) using a set of bases and corresponding coefficients, where the relationship between coefficients and input data can be defined by *pre-defined fixed* or *learned data-dependent* bases. Using pre-defined bases, such as discrete Fourier transform or wavelet transform (WT), is often computationally cheaper. Data-dependent bases, such as vector quantization or singular value decomposition (SVD), require pre-processing time to learn the bases. However, learned bases potentially remove more redundancy from the input data.

While there are many studies on pre-defined bases models in visualization, learned bases models like the higher-order SVD (HOSVD), as analyzed in this paper, only received some attention during the last few years. In this paper, we analyzed and evaluated higher-order extensions of matrix SVD models, so-called *tensor approximation* (TA) models, in the context of volume visualization. While one existing TA model was previously used for volume visualization [29], it was left unexplored whether other TA-based volume models offer even more volume-visualization-friendly properties or features. For this purpose, existing TA models together with a set of new variations are evaluated for their compactness, approximation power and decomposition-reconstruction performance for large-scale volume data visualizations. Moreover, in order to support adaptive approximation quality to highlight and focus on certain feature scales, the truncation properties of the TA models are studied.

2. Related Work and Motivation

2.1. Compact Volume Representation

The available GPU computing power together with advantages in volume ray-casting made DVR the technique of choice for interactive 3D volume visualization [11]. However, huge amounts of volume data need to be processed for each frame to

display the final aggregated 2D image on screen. 3D data acquisition devices can easily produce data sets too large to be visualized at interactive frame rates at a high quality. Therefore, DVR methods working on compressed 3D data sets – *compression-domain DVR systems* – are an ongoing active research topic [6].

In this context, data reduction is important, first, to save storage space at all stages of the visualization pipeline, and second, to reduce transmission time when copying data between different memory layers. Most approaches follow lossy approximation methods as lossless techniques provide limited gains [13]. Furthermore, effective compression-decompression processes to generate and load a compact data representation are often asymmetric. That is, a computing-intense compression is applied in an offline preprocess, while fast decompression is performed in real-time, possibly on the GPU.

Compact data representations are often mathematical transforms that represent the data in a more compact way, i.e., with as few coefficients as possible. Two classes of data decomposition approaches can be applied: one uses pre-defined bases, while the other computes the bases as part of the output. Examples of pre-defined bases in compression-domain DVR are the discrete Fourier transform or the discrete cosine transform (DCT) (both, frequency domain transforms) as well as the discrete WT (frequency domain transform with variable spatial resolution). Examples of learned bases are dictionaries which replace the data by a small set of pre-defined and learned codewords, e.g., including vector quantization or sparse coding. For a detailed analysis and references of such compact models, we refer to [6].

Compression-domain DVR is an ongoing active research area, nevertheless, SVD-like methods have only recently been exploited [6, 28, 29]. One reason is that the extension of the SVD or principal component analysis (PCA) to higher orders, beyond matrices, is not trivial. However, there are ways to apply SVD-like methods to higher order data arrays like 3D volumes.

The higher-order extensions of SVD are summarized under TA, see, e.g., [18]. The matrix SVD exploits the fact that a 2D data matrix can be decomposed into a few highly significant coefficients and corresponding reconstruction basis vectors. Thus, the SVD computes (a) a rank-R decomposition, and (b) orthonormal column vector basis matrices. The extension of this matrix rank truncation concept to higher-orders is not unique and the two properties from the matrix SVD are generalized by different TA models: the Tucker model preserves the orthonormal factor matrices, while the CANDECOMP/PARAFAC (CP) model preserves the rank-R decomposition.

While the Tucker model has been successfully applied to compression-domain DVR [29, 30], it was unexplored whether other TA-based models further optimize or support compression-domain volume rendering even more. Here, we reviewed what TA features and properties are advantageous for compact volume representation and rendering.

2.2. Tensor Approximation Models

A *tensor* is a higher-order generalization of a vector or multidimensional array, where the order of the tensor defines the number of modes (data directions). In TA methods, a multidimensional input data tensor is factorized into a sum of rank-one tensors. This factorization process is known as *tensor decomposition*, while the reverse is the *tensor reconstruction*.

TA has been applied to various visual data sets such as image ensembles (e.g., [15, 23, 36, 38, 42]) and/or for pattern recognition (e.g., [12, 25]). In graphics, tensor decompositions have been used for example for compact global illumination models like bidirectional reflectance distribution functions (e.g., [8, 24, 33]), for bidirectional texture functions (e.g., [32, 37, 40, 41]), or for texture synthesis (see [41]). In DVR only the Tucker model has been applied so far (see [28, 29, 30]).

Besides CP and Tucker models, there are numerous other models available being mostly hybrid variants of CP and Tucker. One class of hybrid models are the so-called *block-diagonal* tensor decompositions (BTDs; see [19]), which produce a core tensor with blocks along the super-diagonal and zeros elsewhere. Additional constraints or imposed properties on the tensor decompositions can lead to many alternatives. This demonstrates the versatility of tensor approximations (e.g., INDISCAL, CANDELINC, DEDICOM), see [18] for details.

In this work, we compared the pure Tucker and CP models as well as existing and newly designed block-based TA model variations with respect to their suitability for compression-domain volume visualization.

2.3. Compact Model Properties

With respect to compact data models for compression-domain DVR, there are certain expectations or hopes we have on those. So far, we discussed the core aspect of reducing the amount of coefficients through data decomposition to faithfully represent and also visualize 3D volumes. However, there are further properties of compact models that are advantageous in direct volume rendering. For example, we prefer compact representations, which allow further transformations to be applied directly on the coefficients. Well-known examples for such transformations are bit-reduction quantization, thresholding or variable-length encodings convenient for a fast reconstruction. Furthermore, we are also interested in knowing the sparsity and distribution of our coefficients to be exploited for compression, such as cluster patterns, banded-ness, ordering, frequency bands, coefficient to signal reconstruction significance levels, or coefficient to signal noise ratios.

Wavelet decompositions are known for generating a highly sparse set of coefficients, with many being zero or insignificantly small, which can adaptively be thresholded dependent on the desired approximation quality. The coefficients of matrix SVD-like approaches are also highly sparse (zeros except for diagonal) and ordered by magnitude. Similarly are the coefficients of the CP tensor model highly sparse (values in the super-diagonal, but zeros elsewhere) and ordered by magnitude. In contrast, the Tucker model generates a dense set of coefficients (a core tensor) without strict ordering (except for the first coefficient being by far the largest one).

Another desired property of a compact data model is its capability to support variable spatial reconstruction resolutions,

as well as extracting data at different frequency or feature-size scales. As defined in [29], we refer to *multiresolution* data and *multiscale* feature reconstruction properties, respectively. WTs are well known for separating their coefficients into low-frequency and high-frequency components which are spatially localized, thus allowing for a spatially varying reconstruction at different frequency scales. Tensor decompositions exhibit similar properties, with the input data being represented in a dimensionality reduced subspace; similar to the SVD matrix decomposition, which allows selecting major features and trends in the matrix data. To achieve this behavior, incremental approaches have been applied to higher-order TA (e.g., [29, 30, 41]) using the Tucker model and higher-order tensor rank truncation. In [29, 30] it has been observed that, for example, WT and TA seem to exhibit different adaptive reconstruction behaviors: by computing the bases as part as the compression output, TA is often able to better capture spatial features at different scales. Furthermore, it has been confirmed in [30, 41] that TAs offer competitive data reduction ratios.

With respect to advantageous compression-domain DVR properties, we evaluated the accuracy and sparseness of truncated TA models. Notably, we compared the compression ratios of similar approximation qualities of the truncated Tucker decomposition (TuD) to the truncated CP decomposition (CPD) and truncated BTDs. Moreover, we conducted a performance analysis of the decomposition-reconstruction process. While we are aware that state-of-the-art volume visualization is performed on the GPU [6], we show our performance analysis only on the CPU. In this work, we aimed to illustrate the main conceptual picture on the computation times without going into too many details on parallel implementation versions of the many presented algorithms. Yet, previous compression-domain approaches have already demonstrated that TA reconstruction is feasible for real-time volume visualization, [28, 29]. Nevertheless, where applicable, we give hints on parallel implementation strategies.

3. Tensor Decompositions

In this section, we outline the basic TA models and their current limitations for volume visualization applications.

3.1. Notation

Regarding notation and symbol style, u will denote a vector, $\mathbf{U}$ a matrix and $\mathscr{A}$ a tensor (this paper will focus only on three-way instances). Their sizes will be denoted with the letter I (or I_n when the object has a different size along each mode n, with I_1 being the first). The symbol $\circ$ will denote the outer product of vectors as defined in [18]: $\mathbb{R}^{I_1} \times \cdots \times \mathbb{R}^{I_N} \rightarrow \mathbb{R}^{I_1 \times \cdots \times I_N}$. A subindex in parentheses as in $\mathbf{A}_{(n)}$ refers to the n-th mode unfolding (or matricization) of a tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ into a concatenation of all its slices, as defined in [17, 20]. For example, the matrix $\mathbf{A}_{(1)}$ of a third-order tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is an unfolded short fat matrix $I_1 \times (I_2 I_3)$, according to [17]. Finally, the Frobenius norm (equivalent to the L_2 norm) will be used in all of these, i.e., for any object X, $\|X\| = \sqrt{\sum_{x_i \in X} x_i{}^2}$.

3.2. CP Model

The basic idea of the CP model is to decompose multidimensional data into a finite number of rank-one tensors. This model was first formulated by Hitchcock [16] and popularized by Carroll and Chang [9] under the name CANDECOMP in psychometrics, and by Harshman [14] under the term PARAFAC. Kiers [17] generalized those models in what we refer now as CANDECOMP/PARAFAC decomposition (CPD). Formally, the CPD is the outer product of N vectors $u^{(1)} \circ u^{(2)} \cdots \circ u^{(N)}$, with $u^{(n)} \in \mathbb{R}^{I_n} \forall n$. Similar to the matrix SVD ($N = 2$), the CPD ($N >= 3$) is a rank-R decomposition, which can be used to approximate to original data by truncating rank-one vectors. The CP model is commonly written as the sum of R weighted rank-one vectors:

$$\mathscr{A} \approx \widetilde{\mathscr{A}} = \sum_{r=1}^{R} \lambda_r \cdot u_r^{(1)} \circ u_r^{(2)} \cdots \circ u_r^{(N)} \tag{1}$$

An exact equality can sometimes be achieved, in which case the tensor is called rank-decomposable; R is then the rank of the tensor, and is possibly bigger than $\max(I_1, I_2, \ldots, I_N)$. The rank decomposition of a higher-order tensor is often unique, but not always [7, 27]. From the r column vectors $u_r^{(n)}$, matrices $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}$ can be formed (where $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R} \forall n$). Likewise, the weights λ_r can be thought as forming one array Λ of coefficients. These weights arise by normalizing the r-th column of these matrices, for $r = 1, \ldots, R$. Fig. 1 illustrates this kind of decomposition in the third-order case.

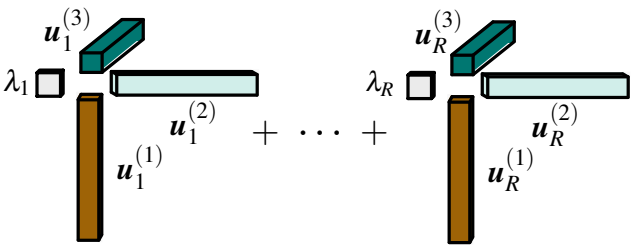


Figure 1: Illustration of $\widetilde{\mathscr{A}} = \sum_{r=1}^{R} \lambda_r \cdot u_r^{(1)} \circ u_r^{(2)} \circ u_r^{(3)}$.

In fact, the CPD is not limited to rank-one, it can also be produced by a rank-R decomposition (see, e.g., [20, 43]). A rank-R CPD can be computed as indicated in the appendix.

3.3. Tucker Model

The TuD was introduced in [34] and mostly popularized in [35], and originally thought as a 3-way generalization of PCA. In posterior applications it has been also viewed as a HOSVD [20]. It factorizes a tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ into one orthonormal basis factor matrix $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ for each mode n of $\mathscr{A}$ and an *all-orthogonal* (see [20]) core tensor $\mathscr{B} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ (illustrated in Fig. 2 for a third-order tensor):

$$\widetilde{\mathscr{A}} = \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_N \mathbf{U}^{(N)} \tag{2}$$

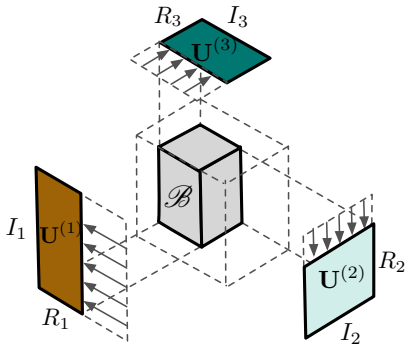


Figure 2: Illustration of $\widetilde{\mathscr{A}} = \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$.

The TuD can be considered as a form of higher-order principal component analysis. In contrast to the 2D case, the TuD of order three or higher is generally not unique. This leaves degrees of freedom that may be exploited whenever specific targeted properties are the goal. For example there is ongoing research [18] on how to transform the core (together with the corresponding factor matrices) to arrive at a more convenient structure and distribution of the core tensor entries. For example, a CPD can be regarded as a special case of the TuD by interpreting the coefficient vector Λ as the super-diagonal of a Tucker core tensor with otherwise zero entries. The problem of turning a full Tucker core into such a CP-shaped object has already been explored before; however, it is a non-trivial matter. While some previous efforts aim to create as many zeros as possible in the core by a number of convenient rotations, a complete super-diagonalization is impossible in general, even in the case of symmetric input data [18]. In practice, as we will show later in this paper, CP needs a much larger number of ranks to attain the accuracy of Tucker.

A Tucker rank-(R_1, R_2, R_3) decomposition can be computed as indicated in the appendix.

3.4. Rank Truncation and its Limits

Usual decomposition algorithms directly produce representations in one of the models reviewed so far. However, one can further modify them: in particular, tensor rank truncation of a decomposition is interesting for the sake of multiscale visualization. More specifically, only a limited number of coefficients may be used for data reconstruction. This makes it possible to reduce the memory footprint and limit reconstruction costs whenever higher efficiency is needed to visualize an approximate 3D volume. Ideally, only the coefficients that account for the least relevant information should be eliminated.

The matrix SVD yields a diagonal matrix whose first coefficients capture most of the decomposition energy, and the first k factors give in fact the best possible rank-k approximation [10]. This inspires a direct approach to perform *tensor rank truncation*. It consists of keeping only the first largest elements of the CP coefficient vector Λ or the top-left-front corner cube of the Tucker core tensor $\mathscr{B}$, and the corresponding factor matrix columns, while cropping the rest. Fig. 3 illustrates this basic tensor rank truncation for the Tucker model.

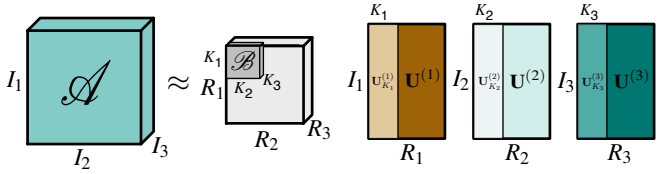


Figure 3: Truncation of a rank-(R_1, R_2, R_3) TuD into a rank-(K_1, K_2, K_3) TA, where $K_n < R_n \, \forall n$.

Notably, while SVD truncation is optimal for matrices, this is not generally the case for tensors of order three or higher. On one hand, imposing all-orthogonality on the TuD's core tensor $\mathscr{B}$, results in well-behaved truncation results in practice [21]. Suter et al. [29, 31] showed that even though the core tensor coefficients are not guaranteed to be in strictly decreasing order (as is the 2D SVD case), progressive rank truncation in the Tucker model works well for adaptive multiscale feature visualization. On the other hand, CPDs are fragile in terms of truncation and are prone to introduce artifacts, especially for lower-rank reconstructions, as illustrated in Fig. 4. However, a CP model can offer magnitude ordering and sparseness of coefficients, e.g., if computed by an incremental tensor decomposition.

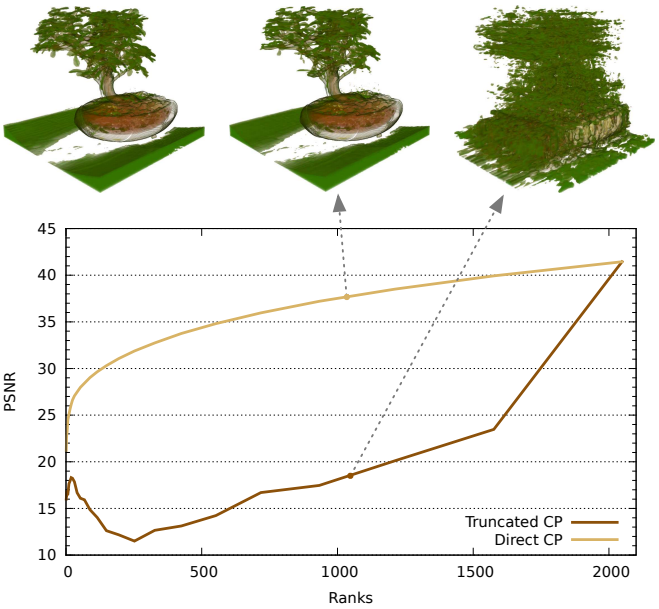


Figure 4: A 256^3 bonsai tree data set (top left) is directly decomposed into a rank-1024 CP (top center) achieving a compression ratio over 1:21. In contrast, truncating a rank-2048 CP to 1024 ranks, produces severe artifacts (top right). The accuracy is expressed by means of the peak signal-to-noise ratio (PSNR). The decompositions are computed with the popular Alternating Least Squares (ALS) algorithm.

4. Incremental Decomposition Models

4.1. Residual Decomposition

As we have just discussed, higher-order tensor decompositions can be ill-conditioned for rank truncation. In order to

further address this, we consider incremental tensor decomposition variants, where the tensor $\mathscr{A}$ is first approximated by a tensor $\widetilde{\mathscr{A}_0}$ that only employs a comparatively small number of coefficients. Then the problem is recursively solved on the residual $\mathscr{A} - \widetilde{\mathscr{A}_0}$, yielding a new set of coefficients and matrix columns for a residual approximation $\widetilde{\mathscr{A}_1}$. The full approximation is eventually given by the sum $\widetilde{\mathscr{A}} = \sum \widetilde{\mathscr{A}_i}$.

We refer to this as residual-based approaches, which are typically organized in a block-diagonal tensor structure. Fig. 5 illustrates this concept: the core tensor is neither a pure Tucker model nor a pure CP model; instead, B blocks $\mathscr{B}_b$ of coefficients are placed along its diagonal. In the extreme case, these blocks can be of size one (see [26, 43]). This approach can conceptually be viewed as a subdivision of the original problem, in which progressively better solutions are attained. The iterative construction guarantees that truncation of blocks results in a progressive approximation. Note that this form of a block-diagonal core tensor offers flexibility with respect to magnitude ordering and sparseness of coefficients.

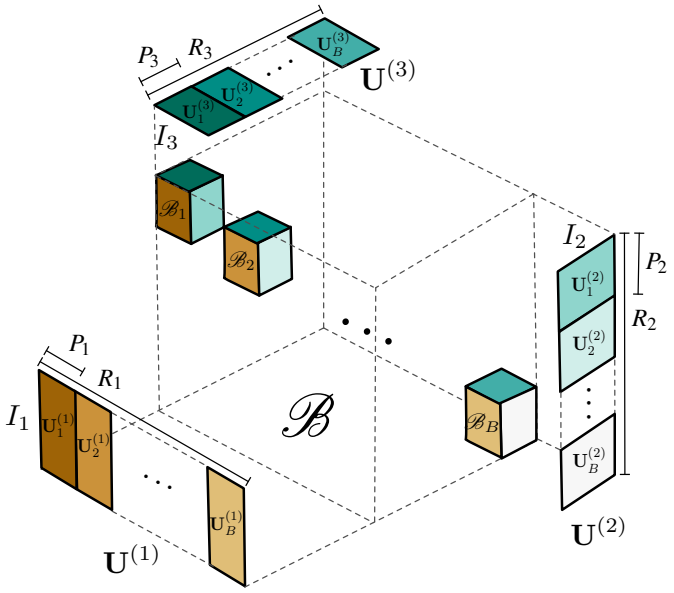


Figure 5: Block TuD: the core tensor entries (i.e., the blocks $\mathscr{B}_1, \ldots, \mathscr{B}_B$) are located along the super-diagonal of the final core tensor $\mathscr{B}$, while the $\mathbf{U}_1^{(n)}, \ldots, \mathbf{U}_B^{(n)}$ are concatenated to form $\mathbf{U}^{(n)}$ for $n = 1,2,3$.

Incremental methods are not the only way to obtain a decomposition in this block-shaped form. For example, de Lathauwer and Nion [22] propose an ALS algorithm for small tensors which, instead of calculating residuals, simultaneously computes all the rank-(P_1, P_2, P_3) blocks of such a decomposition. This approach, however, is computationally expensive and therefore impractical for large tensors (e.g., large volume data sets).

Block-shaped tensor decomposition models, including the plain CP model as a special case, are well suited for parallelization and selective access. Since each core tensor block $\mathscr{B}_b$ and its factor matrices $\mathbf{U}_b^{(n)}$ form an independent reconstruction unit, parallel and selective reconstruction is easily facilitated.

Next, we explore and contribute block-based specific TA models that exploit incremental decomposition and reconstruction strategies.

4.2. Block CP

A simple incremental approach is the block CP decomposition (BCPD). Each of the diagonal blocks contains only entries along the super-diagonal, as in the traditional CP model and shown in Fig.6(a). However, this block-wise disposition is not produced in one step, but rather incrementally forming blocks of successively generated CP coefficients. At each step, the remaining residual tensor $\mathscr{A}'$ is CP-approximated with a vector Λ' and factor matrices $\mathbf{U}'^{(n)}$; these are appended to the (growing) Λ and $\mathbf{U}^{(n)}$, respectively (Alg. 1). After each of the B blocks, the overall approximation improves as the residual keeps getting closer to a null tensor. Because of the nature of CP representations, the coefficients structure of the incremental version is the same as the original as indicated in Fig.6(a). While an incremental decomposition with B rank-one blocks [43] yields in the 2D case (i.e., SVD) the same result as a one-step direct rank-R CP approximation, this is not in general true for higher orders.

Algorithm 1 B-block BCPD with block rank-P and k iterations.

BLOCK_CP($\mathscr{A}, B, P, k$)
$\mathscr{A}' \leftarrow \mathscr{A}$ {initial residual tensor}
$\Lambda \leftarrow 0$ {empty vector}
$\mathbf{U}^{(n)} \leftarrow \mathbf{0}$ {empty matrix, $\forall n = 1,2,3$}
for $b = 1, \ldots, B$ **do**
  $(\Lambda', \mathbf{U}'^{(1)}, \mathbf{U}'^{(2)}, \mathbf{U}'^{(3)}) \leftarrow$ CP_ALS($\mathscr{A}', P, k$) {compute a single rank-P BCPD}
  $\Lambda \leftarrow$ append(Λ, Λ')
  $\mathbf{U}^{(n)} \leftarrow$ append$(\mathbf{U}^{(n)}, \mathbf{U}'^{(n)}) \forall n = 1,2,3$
  if $b < B$ **then**
    $\mathscr{A}' \leftarrow \mathscr{A}' -$ reconstruct$(\Lambda', \mathbf{U}'^{(1)}, \mathbf{U}'^{(2)}, \mathbf{U}'^{(3)})$
  end if
end for
return $(\Lambda, \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)})$

4.3. Block Tucker

Another direct incremental block-based tensor decomposition extension is to use an incremental block TuD (BTuD) model. Using the same principle of recursively approximating the residuals, instead of CP's one-way vector Λ of weights, a $P_1 \times P_2 \times P_3$ core tensor sub-block $\mathscr{B}'$ is added in each step, see also Fig.6(b). We outline this block tensor variant in Alg. 2.

4.4. Block CP-Tucker

As an alternative, we explore a hybrid block CP-Tucker decomposition (BCPTuD) that makes use of an idea proposed in [39]. The idea is to project a tensor onto an axis system (i.e., a number of factor basis matrices) as a way to extract features from the original data. In our case, in each step we obtain a Tucker-shaped core tensor block $\mathscr{B}'$ by projecting the residual data tensor $\mathscr{A}'$ onto the three factor matrices $\mathbf{U}'^{(n)}$ obtained from a CP rank-P decomposition of $\mathscr{A}'$ and discarding the Λ'

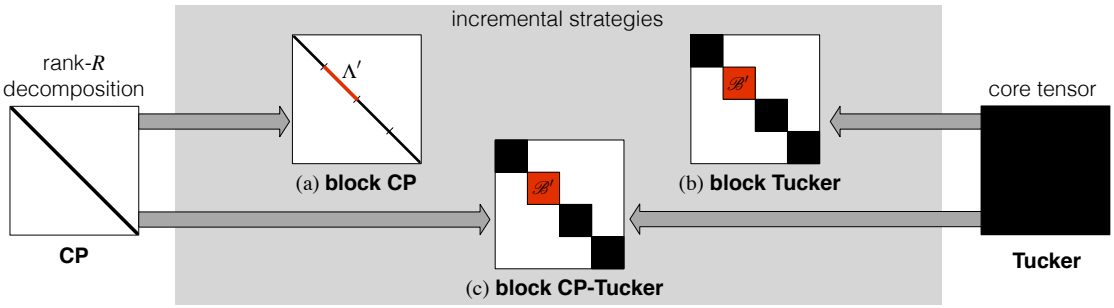


Figure 6: Core tensor coefficient layouts for the different tensor models analyzed in this work, illustrated in 2D. The CP and the Tucker model form the two extremes with respect to the coefficient sparsity.

Algorithm 2 B-block BTuD with multilinear block rank-(P_1,P_2,P_3) and k iterations.

1: BLOCK_TUCKER($\mathscr{A},B,P_1,P_2,P_3,k$)
2: $\mathscr{A}' \leftarrow \mathscr{A}$ {initial residual tensor}
3: $\mathscr{B} \leftarrow (P_1B)\times(P_2B)\times(P_3B)$ zero tensor
4: $\mathbf{U}^{(n)} \leftarrow \mathbf{0}$ {empty matrix, $\forall n=1,2,3$}
5: **for** $b=1,\ldots,B$ **do**
6: $(\mathscr{B}',\mathbf{U}'^{(1)},\mathbf{U}'^{(2)},\mathbf{U}'^{(3)}) \leftarrow$ TUCKER_ALS $(\mathscr{A}',P_1,P_2,P_3,k)$ {compute a rank-(P_1,P_2,P_3) Tucker block}
7: $\mathscr{B} \leftarrow \text{append}(\mathscr{B},\mathscr{B}')$ {add the new block $\mathscr{B}'$ as the next block along the super-diagonal of $\mathscr{B}$}
8: $\mathbf{U}^{(n)} \leftarrow \text{append}(\mathbf{U}^{(n)},\mathbf{U}'^{(n)})\,\forall n=1,2,3$
9: **if** $b<B$ **then**
10: $\mathscr{A}' \leftarrow \mathscr{A}' - \text{reconstruct}(\mathscr{B}',\mathbf{U}'^{(1)},\mathbf{U}'^{(2)},\mathbf{U}'^{(3)})$
11: **end if**
12: **end for**
13: **return** $(\mathscr{B},\mathbf{U}^{(1)},\mathbf{U}^{(2)},\mathbf{U}^{(3)})$

coefficients. This variant has the same outline as the BTuD in Alg. 2, where it suffices to substitute line 6 with the three code lines given in Alg. 3 below, and shares the same core tensor coefficients structure as illustrated in Fig. 6(c).

Algorithm 3 CP-Tucker alternative for one block of rank-(P_1,P_2,P_3) and k iterations. (Substitutes line 6 of Alg. 2)

1: $(\Lambda',\mathbf{U}'^{(1)},\mathbf{U}'^{(2)},\mathbf{U}'^{(3)}) \leftarrow$ CP_ALS $(\mathscr{A}',\max(P_1,P_2,P_3),k)$
2: $\mathscr{B}' \leftarrow \mathscr{A}' \times_1 \mathbf{U}'^{(1)^T} \times_2 \mathbf{U}'^{(2)^T} \times_3 \mathbf{U}'^{(3)^T}$ {get the core $\mathscr{B}'$ from projection of $\mathscr{A}'$ onto CP matrices}
3: $\mathbf{U}'^{(n)} \leftarrow \text{columns}_{1,\ldots,P_n}(\mathbf{U}'^{(n)})\,\forall n=1,2,3$

Starting from the basic CP and Tucker tensor decomposition models, we have outlined their block-incremental extensions, resulting in three different block-based tensor decomposition alternatives as summarized based on their coefficient vector and core tensor layouts in Fig. 6.

4.5. Initialization Choices

The initial variable values that an iterative algorithm relies on can affect its outcome. In our case these are the initial factor matrices $\mathbf{U}^{(n)}$ of the ALS tensor decomposition methods described in the appendix, which may be initialized in several ways [18]. We explore three different methods (the first two are covered in the literature, while we contribute the third one):

- HOSVD: We form the initial $\mathbf{U}^{(n)}$ from the R_n leading left singular vectors of the SVD of the unfolded matrix $\mathbf{A}_{(n)}$ (or, equivalently, the R_n leading eigenvectors of $\mathbf{A}_{(n)}\mathbf{A}_{(n)}^T$). This is only possible if $R_n < I_n\,\forall n$.
- Random: The matrix $\mathbf{U}^{(n)}$ is populated with values that follow a random continuous distribution.
- DCT: Each column of the initial matrices $\mathbf{U}^{(n)}$ is represented by a discrete cosine transform (type II). The entries of the DCT-II matrix are computed as:

$$a_{ij}^{(n)} = C_i \cos\left(\frac{(2(j-1)+1)(i-1)\pi}{2I_n}\right) \qquad (3)$$

where $i \in \{1,\ldots,I_n\}$ and $j \in \{1,\ldots,R_n\}$ (i.e., the last columns of the standard square DCT-II matrix are discarded). The coefficients are set as $C_1 = \sqrt{1/I_n}$ and $C_i = \sqrt{2/I_n}\,\forall i>1$, so that the columns form an orthonormal basis.

5. Space and Time Analysis

As we argued in the introductory sections, storage and data transmission costs play a vital role in interactive visualization applications, such as direct volume rendering. Offline decomposition and online reconstruction times (denoted as T_D and T_R, respectively) are also important, especially the latter. In what follows, we give a theoretical cost analysis of storage and performance for each of the described TA models. From now on we will focus on data sets with symmetric size. Thus I will denote the largest of $\{I_1,I_2,I_3\}$. Regarding the number of ranks, we will use R (or P for block methods): in Tucker-based models, R (or P) will denote the largest of $\{R_1,R_2,R_3\}$ (or of $\{P_1,P_2,P_3\}$).

5.1. Space Complexity

In order to assess the storage cost of a decomposition, we count its *number of non-zero coefficients* (NNC). Further quantization and variable length encoding of coefficients is left out for the sake of the limited scope. The NNCs required for a

third-order decomposition of each model is summarized in Table 1. The summands in every expression take into account the three factor matrices and either the array of coefficients (CP models, rows 1 and 2) or the core tensor (Tucker models, rest of the rows). In terms of space needed, the B-block variants imply the corresponding multiple of one reduced size block's requirements.

As a significant observation, CP and Tucker show very different rank-to-NNC ratios. In particular, if we consider a rank-R_{CP} CP and rank-(R_T, R_T, R_T) Tucker approximation of a given tensor $\mathscr{A} \in \mathbb{R}^{I \times I \times I}$, for an equal NNC the following equality must hold:

$$(3I+1) \cdot R_{CP} = 3I \cdot R_T + R_T^3 \tag{4}$$

If $3I \cdot R_T \ll R_T^3$, the size of the Tucker factor matrices is small compared to that of the core tensor and then we have $R_{CP} \propto R_T^3$. It follows that we typically have $R_{CP} > R_T$ for equal NNCs. Thus CP stores the bulk of its coefficients in wide factor matrices $\mathbf{U}^{(n)}$ of size $I_n \times R_{CP}$, whereas the Tucker model has most coefficients in its core tensor $\mathscr{B}$ of size R_T^3 (as it can be seen in Fig. 1, cf. Fig. 2).

5.2. Time Complexity

Here we provide an analysis of the asymptotic time cost for each algorithm. The complexity is studied in terms of the largest contributions of the two most important variables, I and R. In general, for a similar performance, different models need different numbers of ranks. Thus the terms I and R (or P) have, for practical applications, distinct relative weights. For example, for low CP compression ratios and large enough input volumes, frequently $R > I$.

We analyze both the decomposition and reconstruction procedures; the former is divided into a) initialization cost T_I, and b) algorithm run-time cost T_D. We study the reconstruction of the volume as a whole, so that we do not consider on-demand reconstruction of individual voxels only when they are needed (e.g., in a rendering stage). The reason is two-fold: the whole volume reconstruction is always at least as fast as the other alternative (much faster in the case of Tucker, as we argue in section 5.2.3); and the additional storage space that it requires can be minimized as much as needed by partitioning the volume decomposition into many small enough subregions (as it is done in [29]).

The basic CP rank-R and Tucker rank-(R,R,R) decomposition algorithms are given in the appendix, while the block variants have been introduced in the previous section.

5.2.1. Initialization

At the start, the factor matrices must be populated. We estimate the initialization times T_I for each matrix $\mathbf{U}^{(n)}$ as follows:

- HOSVD: This comprises a matrix-matrix multiplication $\mathbf{A}_{(n)}{\mathbf{A}_{(n)}}^T$, cost $O(I^4)$, and an eigenvector computation of the $I \times I$-sized result, cost $O(I^3)$.
- Random and DCT: $O(I \cdot R)$, as each entry has a constant cost.

Since a single ALS step relies on two matrices, this initialization procedure is needed only for two out of the three matrices. Initialization costs can be calculated beforehand just by knowing the data size and the number of ranks, hence the most expensive matrix can be dropped by permuting the modes.

5.2.2. Decomposition

After initialization, we analyze the asymptotic costs of the iterative algorithms:

- Rank-R CP: The workload is dominated by the first of the matrix products, cost $O(I^3 \cdot R \cdot k)$ (line 7 of CP-ALS, in the appendix), and the pseudo-inverse computation of $\mathbf{V}^\dagger$, cost $O(R^3 \cdot k)$, on the same line. Since the CP-Tucker relies on CPDs, its cost per iteration is analogous.
- Rank-(R,R,R) Tucker: The most expensive steps are the first tensor times matrix product, cost $O(I^3 \cdot R \cdot k)$ (line 5 of Tucker-ALS, in the appendix), and the last tensor times matrix product over R, cost $O(I \cdot R^3)$ (line 9 of Tucker-ALS).
- Incremental block variants (Algs. 1,2,3): As in the space complexity analysis, each block has the same size and demands an equivalent computational effort. Hence the total time is upscaled by B, but for smaller ranks $P \ll R$ per block. Additionally, $B-1$ reconstructions must be additionally computed in order to handle the calculation of residuals.

The costs are derived in greater detail in the supplementary material, and summarized in Table 2 (middle column).

5.2.3. Reconstruction

The asymptotic time complexities for the reconstruction steps are analyzed next.

- Rank-R CP: Each rank component requires an outer product over three vectors, which takes $O(I^3)$ operations. When the R terms are considered, the total cost amounts to $O(I^3 R)$.
- Rank-(R,R,R) Tucker: A naive element-wise reconstruction needs $O(I^3 \cdot R^3)$ operations and is not the most efficient approach. Instead, the resulting tensor $\widetilde{\mathscr{A}}$ can be calculated as

$$\widetilde{\mathscr{A}}[i_1,i_2,i_3] = \sum_{r_1=1}^{R} \mathbf{U}^{(1)}[i_1,r_1] \cdot \mathscr{B}''[r_1,i_2,i_3] \tag{5}$$

where

$$\mathscr{B}''[r_1,i_2,i_3] = \sum_{r_2=1}^{R} \mathbf{U}^{(2)}[i_2,r_2] \cdot \mathscr{B}'[r_1,r_2,i_3] \tag{6}$$

and

$$\mathscr{B}'[r_1,r_2,i_3] = \sum_{r_3=1}^{R} \mathbf{U}^{(3)}[i_3,r_3] \cdot \mathscr{B}[r_1,r_2,r_3], \tag{7}$$

which accounts for $O(I^3 \cdot R) + O(I^2 \cdot R^2) + O(I \cdot R^3)$ operations.

Decomposition model	NNC
rank-R CPD	$(3I+1)\cdot R$
B-block BCPD, rank-P each	$B\cdot(3I+1)\cdot P$
rank-(R,R,R) TuD	$3I\cdot R+R^3$
B-block BTuD, rank-(P,P,P) each	$B\cdot(3IP+P^3)$
B-block BCPTuD, rank-(P,P,P) each	$B\cdot(3IP+P^3)$

Table 1: Worst-case storage requirements (i.e., maximum number of non-zero coefficients) for each of the analyzed decomposition models.

- Incremental variants: The block based approaches are derived analogously with fewer ranks $P \ll R$ per block, but with a scale factor B.

The costs are summarized in Table 2 (right column).

6. Experimental Performance

After giving theoretical estimations, we measure empirical performance under a number of different variable parameters. To this end, the presented tensor decomposition models were tested over several 3D tensors, i.e. volume data sets. The main goals of this section are a) to check which performance (in terms of accuracy and initialization, decomposition and reconstruction times) can be expected when current TA models are applied to real-world volume visualization; b) find out what parameters have the most impact and relevance for that purpose; c) assess which TA decomposition algorithms work best under these variables; and d) measure how robust TA methods are with respect to rank truncation.

6.1. Setup

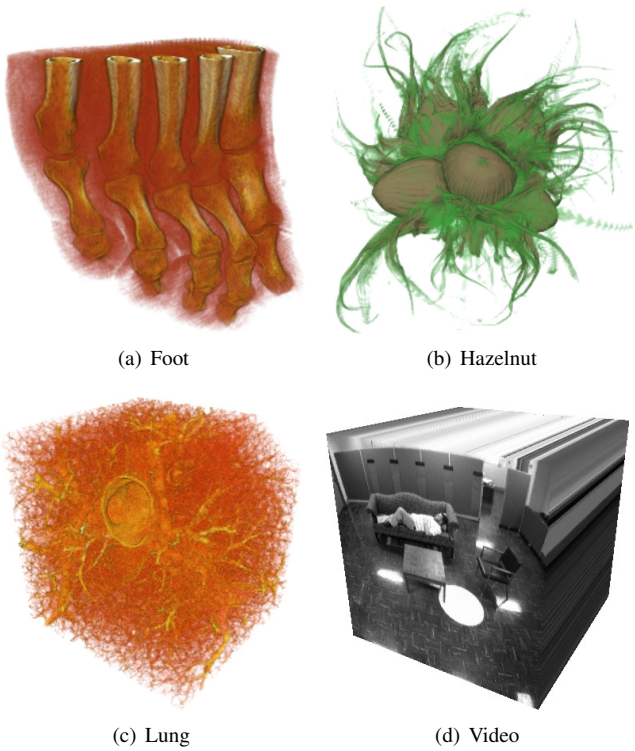
In this section we describe the whole parameter range we cover, how our simulations are performed and which assumptions are made throughout our experiments.

6.1.1. Data Sets

We tested the following 8-bit volume data sets (see Fig. 7):

- A rotational C-arm X-ray 256^3 scan of a human *Foot*, which includes tissue and bone.
- A micro-computed tomography 512^3 scan of *Hazelnuts*, de-noised by post-processing.
- A 1024^3 phase-contrast synchrotron tomography scan of a rat *Lung*.
- A raw grayscale *Video* of a person moving around a laboratory room, converted to a collection of 256 images sized 256^2 pixels each.

These choices aim to capture performance dependence on the amount of zero values in the data, as well as on its degree of redundancy. For example, the video does not feature a change of scene, and the view point is fixed. This makes the data very repetitive along the third (temporal) axis, making it a good example use case for analyzing TA redundancy detection.



(a) Foot (b) Hazelnut

(c) Lung (d) Video

Figure 7: Volume data sets used for the TA models tests.

6.1.2. Software and Hardware

We ran our simulations on a 16-core Intel Xeon CPU E5-2670 with 2.60GHz and 32GB of RAM. All algorithms were implemented in two different toolboxes:

- A tensor approximation extension of vmmlib (a vector and matrix math library [2]) in C++. OpenMP was used to take advantage of multicore processing capabilities.
- The MATLAB Tensor Toolbox [5], with its dense tensor capabilities detailed in [4]. MATLAB's default multithreading and just-in-time acceleration were used.

Both vmmlib and MATLAB take advantage of BLAS and LAPACK, high-performance linear algebra libraries for matrix-vector and matrix-matrix operations and transformations. They use block-based algorithms for handling matrices, exploiting CPU parallelism and high cache-based performance.

6.1.3. Parameters and Fitness Measure

As we have stated, the block variants can be regarded as an abstraction of the two basic models. When testing them, we

Decomposition model	T_D	T_R
rank-R CPD	$O(I^3 \cdot R \cdot k) + O(R^3 \cdot k)$	$O(I^3 \cdot R)$
B-block BCPD, rank-P each	$O(B \cdot I^3 \cdot P \cdot k) + O(B \cdot P^3 \cdot k)$	$O(B \cdot I^3 \cdot P)$
rank-(R,R,R) TuD	$O(I^3 \cdot R \cdot k) + O(I^2 \cdot R^2 \cdot k)$	$O(I^3 \cdot R) + O(I \cdot R^3)$
B-block BTuD, rank-(P_1,P_2,P_3) each	$O(B \cdot I^3 \cdot P \cdot k) + O(B \cdot I^2 \cdot P^2 \cdot k) + O(B \cdot I \cdot P^3)$	$O(B \cdot I^3 \cdot P) + O(B \cdot I \cdot P^3)$
B-block BCPTuD, rank-(P_1,P_2,P_3) each	$O(B \cdot I^3 \cdot P \cdot k) + O(B \cdot P^3 \cdot k) + O(B \cdot I \cdot P^3)$	$O(B \cdot I^3 \cdot P) + O(B \cdot I \cdot P^3)$

Table 2: Decomposition T_D and reconstruction T_R time complexity for each of the models. In Tucker-based models, R means $\max(R,R,R)$, and P means $\max(P_1,P_2,P_3)$.

cover the whole range between one single block ($B = 1$) and many rank-$(1,1,1)$ blocks. However, to keep a low parameter dimensionality, only cubic blocks with $P = P_1 = P_2 = P_3$ are considered. When the total number R of desired ranks is not an exact multiple of P, we use a smaller last block, namely one of size $R \mod P$. All plots were generated from a 256^3 centered sub-brick of the Lung data set (except where stated otherwise), and the hosvd() ALS initialization is used.

In the literature, ALS algorithms often stop either when a maximum number of iterations k is reached, or when the fitness quality ceases to improve significantly. However, while for the TuD the convergence can be checked based on the vanishing improvements in the Frobenius norm, for the CPD convergence has to be measured by reconstructing the current model and comparing it to the original data for each iteration. Including regular reconstructions, thus, obfuscates the decomposition timing itself. Therefore, in order to better study the actual decomposition timing, we only employed the first criterion here. Thus we picked $k = 3$ and $k = 20$ for all simulations (except where stated otherwise) for the CP-ALS and the Tucker-ALS algorithms, respectively. Fig. 8 shows an example of how the accuracy evolves during a tensor decomposition, and supports these choices. Regarding the incremental variants, the number of needed iterations does not vary significantly between blocks and can therefore be seen as a particular case of the single-block methods.

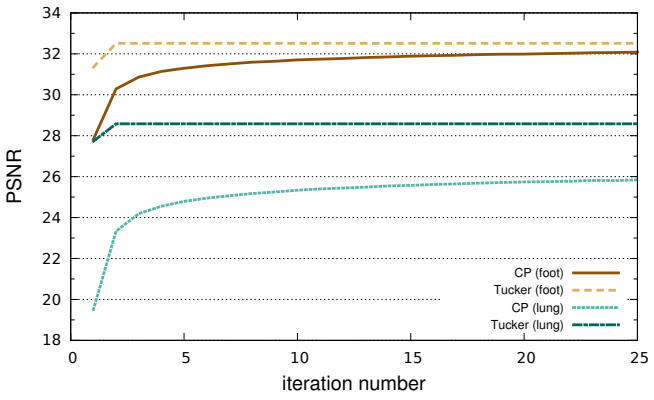


Figure 8: The reconstruction PSNR (compression ratio 1 : 25, rand() initialization method) increases monotonically with the number of iterations. A data set with many zero or almost zero values (the Foot, which has an empty surrounding area) and one with much fewer zeros (the Lung) are tested. In practice, a safe degree of convergence is typically attained after at most $k = 20$ and $k = 3$ iterations in CP-ALS and Tucker-ALS, respectively.

As for measuring the error of an approximation $\widetilde{\mathscr{A}}$ over the input data $\mathscr{A}$, we make use of its difference norm $\varepsilon = \|\widetilde{\mathscr{A}} - \mathscr{A}\|$. We actually express this error as a the PSNR, which is $10 \cdot \log_{10}(255^2/MSE)$ for 8-bit data. $MSE = \varepsilon^2/S$ is the mean squared error, where S is the total number of voxels in the data set.

6.2. Decomposition and Reconstruction

We show the behavior of every model discussed so far in terms of several relevant traits (rate distortion, PSNR, processing times T_D and T_R) in Table 3. We employ one data set of each kind: with many zeros (the Foot), no zero-filled regions (the Lung), and time-dependent (the Video).

In the following vmmlib-generated plots (Figs. 9 for PSNR, 11 for T_D and 12 for T_R), we show how these traits evolve as we increase the compression ratio (the number of ranks for the decomposition is chosen to match the desired ratio in each case). This is referred to as a *forward procedure*. A visualization example for a more realistic compression ratio (1:8) and high accuracy is displayed in Fig. 10, clearly favoring the TuD.

As it is shown in Table 3, the TuD is leading in most aspects over CP. The Tucker model shows compact models at a high approximation quality. However, a better reconstruction quality is achieved with CP in the case of the Video, and we further illustrate this by showing several time slices from it in Fig. 18. Our intuition for CP's superiority in this case is as follows: The Video's low data variation along the z-axis (time) causes overly redundant 3rd factor matrices (see also Table 13) in both algorithms and directions (vertical and horizontal). Horizontal redundancy causes the Tucker core (by construction) to be redundant along the 3rd mode, which is inefficient in terms of space. On the other hand, CP treats each triad of factor matrix column vectors independently and thus the redundancy only affects 1 out of 3 factor matrices.

With respect to performance, we can observe that the basic CPD and TuD are generally faster to generate than their block-based counterparts. For the reconstruction, however, the BCPD is equally fast, unlike the Tucker models where the block-based versions are slower. Hence the flexibility and independence of core tensor blocks and corresponding groups of factor matrix columns can be achieved by a small performance trade-off. The BCP has other benefits as exposed in other experiments and discussed in the conclusion.

In order to put tensor-based data compression performance in perspective with respect to other methods, we have measured the PSNR of a number of wavelet-based compression schemes,

			CPD	BCPD	TuD	BTuD	BCPTuD
blocks × ranks			$1\times(220)$	$4\times(55)$	$1\times(50,50,50)$	$4\times(27,27,27)$	$4\times(27,27,27)$
NNC			169180	169180	163400	161676	161676
Foot	PSNR		29.2308	28.8687	29.7816	29.7210	29.0279
	T_D (s)	vmmlib	70.7839	111.1355	2.3931	8.5955	74.9399
		Tensor Toolbox	46.9849	106.7844	1.4152	9.4137	55.9399
	T_R (s)	vmmlib	3.6194	3.6219	0.3501	0.5645	0.5071
		Tensor Toolbox	32.3390	31.6588	0.3300	0.5074	0.4837
Lung	PSNR		20.6901	20.0028	23.7792	22.1662	21.0328
	T_D (s)	vmmlib	70.5657	111.9481	2.3972	8.6377	75.5489
		Tensor Toolbox	46.8825	109.3242	1.4480	8.8411	56.6315
	T_R (s)	vmmlib	3.6510	3.6365	0.3027	0.5608	0.4849
		Tensor Toolbox	32.0706	31.9553	0.3478	0.4612	0.4632
Video	PSNR		34.6866	33.3837	31.9531	33.0739	32.3678
	T_D (s)	vmmlib	70.0069	111.9747	2.4012	8.6956	74.5006
		Tensor Toolbox	46.7998	112.8838	1.2504	8.8206	54.6573
	T_R (s)	vmmlib	3.6718	3.6042	0.3096	0.5615	0.5606
		Tensor Toolbox	31.6144	32.2113	0.3497	0.4524	0.4602

Table 3: Numerical results (PSNR and decomposition and reconstruction times) for every approximation scheme, evaluated over three data sets. The numbers of blocks and ranks were chosen so that their compression ratios are as similar as possible ($\sim 1:100$). The factor matrix initialization function is hosvd(). The PSNR accuracy values were calculated with vmmlib; Tensor Toolbox results differ always by less than 1%.

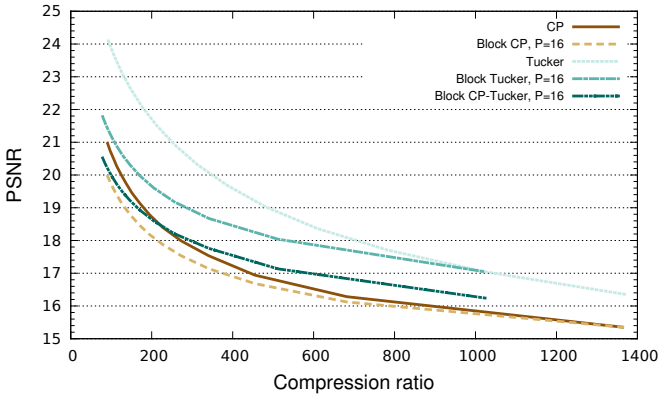
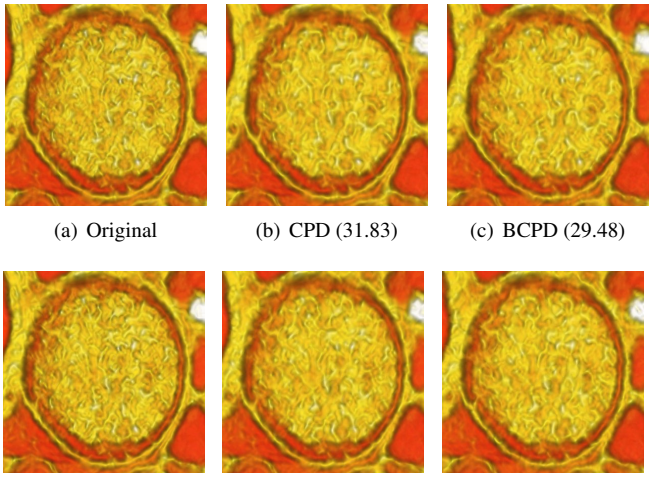


Figure 9: PSNR for different TA models.



(a) Original (b) CPD (31.83) (c) BCPD (29.48)

(d) TuD (32.39) (e) BTuD (29.97) (f) BCPTuD (28.92)

Figure 10: Sectional view of a lung's tube, at a realistic compression ratio (1:8); in parentheses, the PSNR of the volume. The TuD shows a high fidelity, while the other models lose detail to a variable extent.

including well-known wavelets such as the Haar wavelet and the JPEG2000 lossy compression wavelet (MATLAB biorthogonal 4.4 filter). Figure 14 illustrates the PSNR vs. NNC results of a selection of wavelets and the Tucker model. The WT coefficients are hard-thresholded to attain the desired space reduction. The compression ratio is always measured in terms of NNC compared to the initial size; however, we do not consider the additional space needed to encode the sparse coefficient positions after thresholding in the WT domain.

6.3. Effects of Rank Truncation

Given a tensor approximation, we additionally want to study how it is affected by progressive rank truncation. We do this by comparing the resulting approximation accuracy with that of an equivalent (i.e. with the same compression ratio) *direct* tensor decomposition, which serves as a baseline reference. To study this, we employ a *backward procedure*: a single large decomposition is obtained first and is then progressively truncated. Since truncating an already existing decomposition is very fast compared to computing a new one, the main computational load lies in reconstructing the truncated structures.

Fig. 4 shows some example visual artifacts that are generated when truncating different approximation schemes. In Figs. 15, 16, and 17, the approximation results for each of the block-based variants is plotted. The regular non-block variants are also included, since they are equivalent to the special cases where $P = R$. The initial decompositions give rise to the rightmost point of each plotted line, while their successive trun-

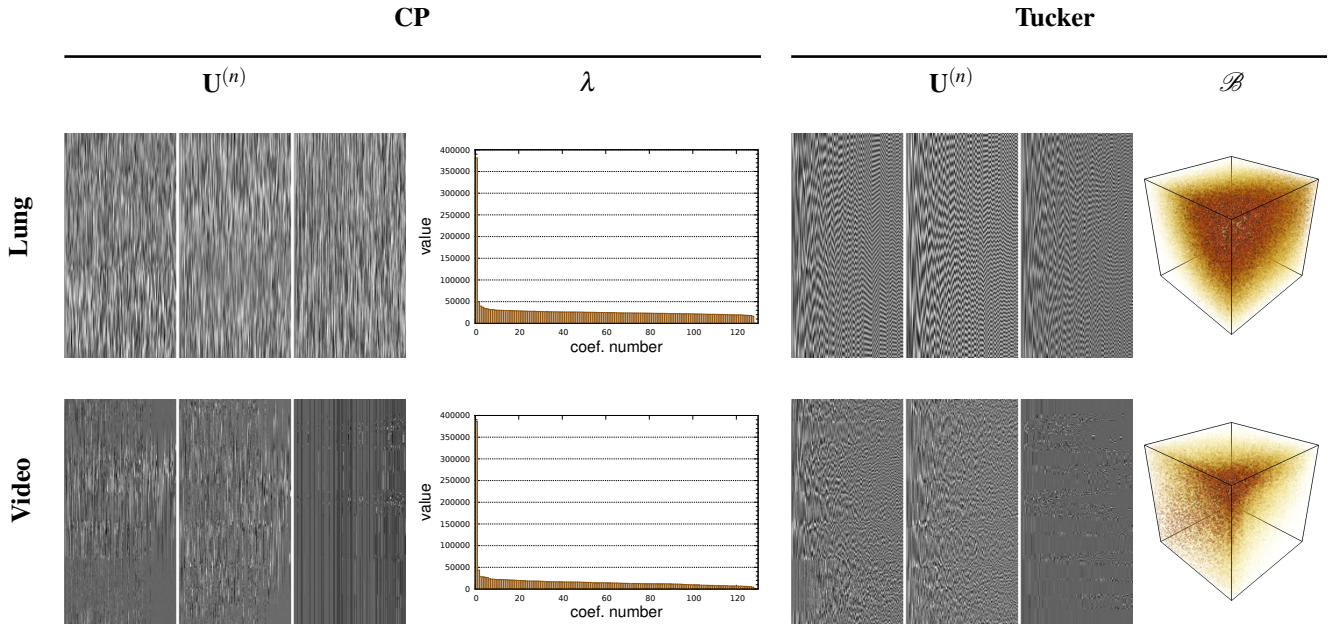


Figure 13: Resulting factor matrices for two different volumes (with $I_n = 256 \forall n$), as well as arrays λ and cores $\mathscr{B}$ for CP and Tucker, respectively. Both ALS algorithms were hosvd()-initialized, and $R_{CP} = R_T = 128$. The matrix images are normalized here in order to ease appreciation. In the Video case, redundancies along the z-axis spread through the third (rightmost) matrices. Furthermore, because of the horizontal redundancy within these matrices (i.e., similarity between columns), most of Tucker's core XY slices are close to zero throughout large areas. This space inefficiency explains its worse performance over this data set when compared with CP, in terms of space versus reconstruction quality.

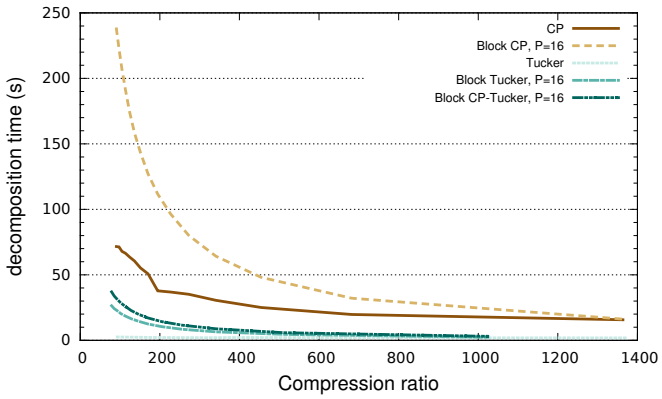


Figure 11: Decomposition times T_D for different TA models.

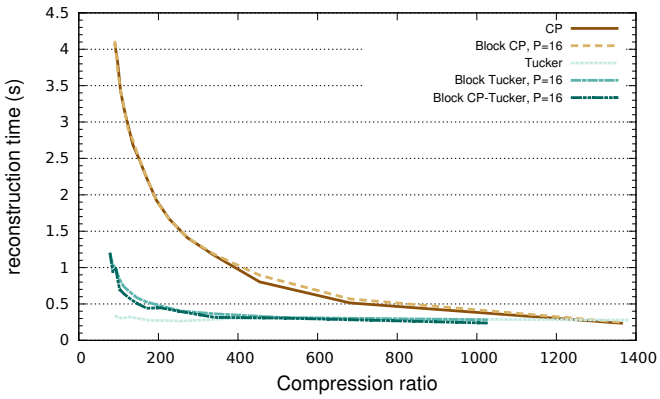


Figure 12: Reconstruction times T_R for different TA models.

cations correspond right-to-left to the remaining data points. Additionally, the direct forward procedure discussed above is shown as well for reference in each case.

We can see in Fig. 15 that for the CP models, the block-based variants can significantly improve the progressive rank truncation behavior. For the TuD (Fig. 16), all block-based models exhibit a nice progressive behavior, but only larger blocks achieve an approximation quality close to the basic Tucker model. The CP-Tucker variants in Fig. 17 show consistent progressive truncation results, albeit at generally lower PSNR.

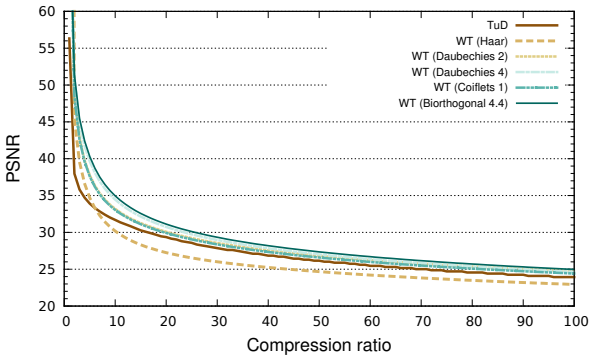
6.4. Initialization Method and Data Size

We also measured how the initialization method affects the compression accuracy and the initialization time T_I of rank-R CPD and rank-(R,R,R) TuD ALS, and show the results in Table 4. The tests were done using the Hazelnut data set. For the rand() initialization choice, 10 experiments were performed and averaged. The results show that the initialization method affects the resulting PSNR and, especially, the initialization time T_I. The method hosvd() takes longer, but in terms of compression quality is approximately equivalent to doing one iteration more.

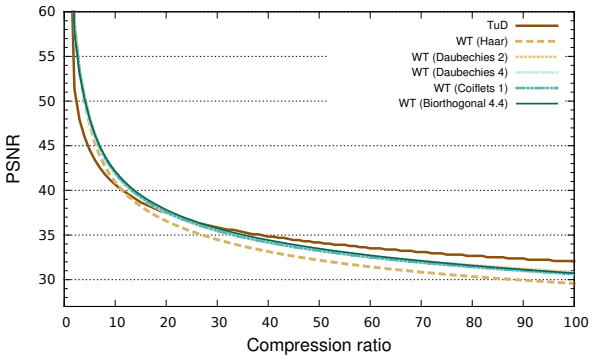
Finally, we address time dependency on data size by approximating different downsampled versions of the Lung data set (Table 5) while keeping constant every other variable. The chosen downsampling method was the popular Catmull-Rom cubic interpolation. The resulting times grow faster than the input data size because both I_n and R_n or P_n have to increase in order to keep the compression ratio constant, agreeing with

		CP, $k = 20$			**Tucker,** $k = 1$		
Compression ratio		$\sim 1:1024$	$\sim 1:512$	$\sim 1:256$	$\sim 1:64$	$\sim 1:16$	$\sim 1:4$
R		86	171	342	124	201	321
PSNR	hosvd()	33.0578	34.6288	36.5028	44.0697	48.7363	53.5358
	rand()	32.9636	34.5489	36.4244	44.0697	48.7363	53.5358
	rand(), $k = k+1$	32.9727	34.5633	36.4393	44.0697	48.7363	53.5358
	dct()	33.0192	34.6009	36.5107	44.0697	48.7363	53.5358
	dct(), $k = k+1$	33.0285	34.6124	36.5248	44.0697	48.7363	53.5358
T_I(s)	hosvd()	20.6663	20.6044	20.6169	20.6587	21.7323	20.5924
	rand()	0.0026	0.0054	0.0109	0.0059	0.0092	0.0145
	dct()	0.0038	0.0075	0.0150	0.0062	0.0092	0.0139
$T_I + T_D$(s)	hosvd()	268.2065	422.4094	689.2532	51.9142	72.0105	111.3845
	rand(), $k = k+1$	259.5677	421.9379	701.3301	41.2930	66.6661	120.9269
	dct(), $k = k+1$	259.3730	422.0736	702.0151	40.9097	66.5039	121.0973

Table 4: Impact of the initialization method on rank-R CPD and rank-(R,R,R) TuD ALS.



(a) Lung



(b) Video

Figure 14: Tucker reconstruction quality compared to different WTs on the Lung and the Video data sets: Haar, Daubechies of order 2 and 4, coiflets of order 1 and biorthogonal wavelets of order 4.4.

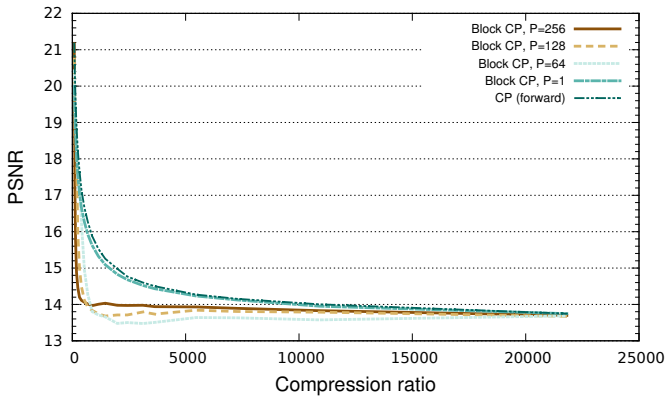


Figure 15: PSNR from progressive truncation of rank-P BCPD for varying P, compared with forward direct CPD calculated for $R = 1$ up to 256.

I^3		32^3	64^3	128^3	256^3	512^3
T_I (s)	CP	0.0032	0.0231	0.1865	2.0561	20.8105
	Tucker	0.0038	0.0255	0.1697	2.0602	20.6484
T_D (s)	CP	0.0187	0.1266	2.6026	36.2253	673.3444
	Tucker	0.2103	0.2567	0.4477	1.9977	17.8453
T_R (s)	CP	0.0028	0.0075	0.0553	1.4473	46.0846
	Tucker	0.0338	0.0363	0.0641	0.2663	2.7099

Table 5: TA timings with increasing tensor sizes: results for CPD-ALS and TuD-ALS (using vmmlib and hosvd() initialization) over the Lung data set, downscaled to several data sizes. The compression ratio is always $\sim 1:250$.

theoretical estimations. The reconstruction PSNR values are not displayed, since they were found to be highly dependent on the downsampling method and do not show a clear pattern.

7. Discussion

With respect to our experimental results, we discuss the following relevant observations:

- In most cases, Tucker offers a superior performance with respect to other TA approaches in terms of both quality and time. It is also competitive when compared to wavelet-based volume compression algorithms. Furthermore, it allows for coefficient reduction in a straightforward fashion by truncation that, as a side benefit, reduces the necessary reconstruction time. Its core tensor values remain arranged in compact form even after coefficient reduction, as opposed to the thresholding strategies on which other compression methods rely. This compact data layout is convenient for storage and allows for

(a) Original (b) CPD (c) TuD

(d) BCPD, B = 4 (equal to truncated BCPD) (e) Truncated CPD (f) Truncated TuD

Figure 18: Time slices from reconstructions of the Video, a data set that is highly redundant along the z-axis. We use a high compression ratio ($\sim 1:168$) to make the differences more appreciable; truncated versions were first compressed to $\sim 1:84$. CP performs best except when truncated. To overcome this its incremental variant is needed, which is still visually cleaner than Tucker-based models.

contiguous memory access, resulting in efficient memory management and reconstruction. In contrast, sparse data decomposition approaches such as WT have a more complex data access pattern during reconstruction.

- Although the proposed incremental variants are the only considered strategies that guarantee by construction a safe truncation, Tucker truncation yields usually the same PSNR as the direct forward procedure; this fact renders it one of the most robust decomposition models. On the other hand, single-block BCPTuD is particularly fragile in that sense.

- Direct forward CPD is prone to severe progressive truncation errors and artifacts. In contrast, B-block BCPD greatly ameliorates this behavior: the more blocks are employed, the better conditioned the result is for truncation (i.e., no significant loss of quality over the forward procedure). Moreover, the response is satisfactory even when the amount of truncated ranks does not correspond to an integer number of blocks.

- Theoretical estimations for decomposition and reconstruction times have been validated. Since in our experiments $R_n, P_n < I_n \forall n$, all terms are dominated by the $O(I^3 \cdot R)$ or $O(I^3 \cdot P)$ contribution. As a consequence, there is indeed a linear time dependency on the number of ranks.

- The TA performance is variably influenced by the input data set, implementation language and parameters. Notably, some decisions do not significantly affect certain dependent variables (see Table 6). Some data sets are more repetitive along one axis than along the others. In these, TA techniques are able to successfully exploit higher redundancy and achieve superior compression quality when compared to less repetitive data sets. This applies especially to CP-based decompositions, to the point of surpassing Tucker in the Video example.

- TuD is typically faster than CPD for both decomposition and reconstruction. In the former, the relation $R_{CP} \propto R_T^3$ puts CP in disadvantage (see Table 2). In the latter, the Tucker reordering strategy detailed in Section 5.2.3 is a

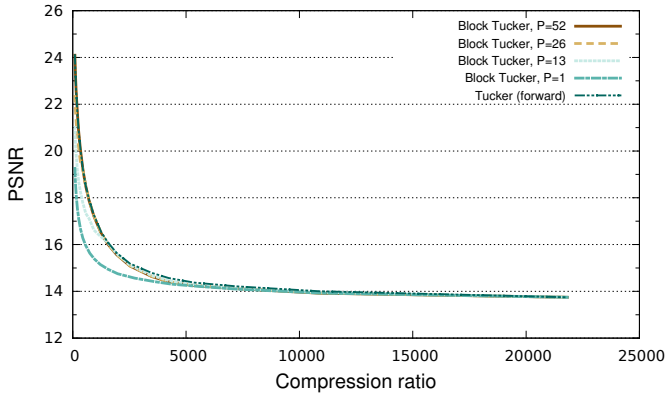


Figure 16: PSNR from progressive truncation of rank-(P,P,P) BTuD for varying P, compared with forward direct TuD calculated for $R_n = 1$ up to 52.

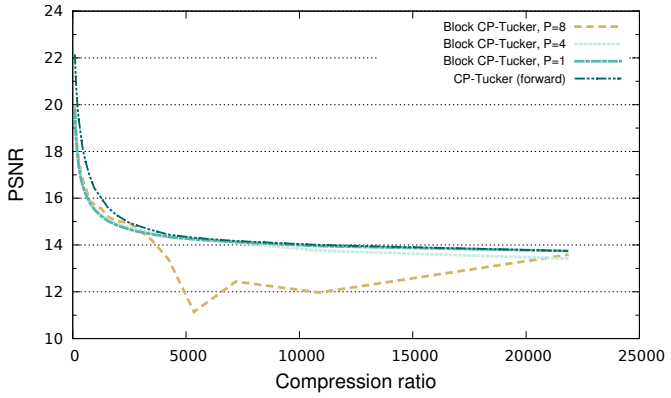


Figure 17: PSNR from progressive truncation of rank-(P,P,P) BCPTuD for varying P, compared with forward direct CPTuD calculated for $R_n = 1$ up to 52. Backward decompositions for $P > 8$ yield too large errors under large truncation and are thus not displayed.

decisive optimization. Even if one considers CP as a Tucker model with a super-diagonal core that strategy would not help because of this rank imbalance: the cost for CP would be $O(I^3 R_{CP}) \propto O(I^3 R_T^3)$, against Tucker's $O(I^3 R_T)$. The CP-Tucker variant fails to enhance either timing aspects, as well as the approximation quality. In practice, the reconstruction for an interactive visualization system is typically implemented in parallel directly on the GPU (see e.g. [28, 29]).

- In agreement with [40], we found no major differences in accuracy regarding the initialization method; the timing comparison depends on the number of ranks. However, hosvd() cannot be used when $R > I_n$ for some n, a frequent case in CPDs.

- The comparison between vmmlib and the Tensor Toolbox shows that the former is normally slower for decomposition, but often faster at reconstruction, which is more important for interactive visualization. Tucker-based algorithms decompress the data by unfolding their compact core (matricizing it) so that matrix products and transformations can be efficiently handled by BLAS and LAPACK. Both vmmlib and MATLAB use these libraries, which explains the similar reconstruction performance. On the other hand, no such optimizations can be used for CP (each element in a CPD gets used only once). We reason that CP is a more iterative model in essence, and therefore performs faster in C++. Besides, C++ is in general much better suited for integrating TA models into an interactive volume rendering system.

Independent variables \ Dependent variables	T_I	PSNR	T_D	T_R
Data set		×		
Data size	×	×	×	×
init()	×	×		
Language	×		×	×
Decomposition algorithm	×	×	×	×
Number of ALS iterations		×	×	
Compression ratio	×	×	×	×

Table 6: Dependency relations: a cross in a cell indicates that altering its row variable (considering that the variables from all other rows are fixed) significantly affects its column variable.

8. Conclusions

Tensor approximation models have previously been demonstrated to be a good alternative for compression-domain direct volume rendering in [28, 29], introducing handles for explorative multiscale feature visualization. Moreover, as for other compression-domain volume representations, TAs online reconstruction causes only little overhead, exploiting view coherence, GPU acceleration as well as caching. Therefore, TA allows for efficient interactive volume rendering (see also [6]).

In this work, we explored the applicability of tensor approximation for 3D volume visualization by considering several tensor decomposition variants. CP is studied in addition to the previously used Tucker model, as well as hybrid versions. We answered rank truncation concerns by showing that incremental decomposition approaches are a viable strategy for achieving robustness. Relevant parameters for every model were identified, and their influence on the considered methods assessed. Both theoretical estimations and empirical measurements proved that the TuD is a superior choice whenever reconstruction time is the critical factor, what is usually the case in interactive 3D visualization applications. Nevertheless, CP can be better in terms of approximation quality for certain data sets, namely those containing a high degree of redundancy. In addition, we showed that incremental CPDs are furthermore guaranteed by construction to be truncatable, thus retaining potential for adaptive reconstruction of highly redundant data. Hence, a possible future line of work is to further improve the applicability of CP-based models for visualization purposes by incorporating the latest, state-of-the-art parallel decomposition approaches.

Acknowledgements

This work was supported in part by the Forschungskredit of the University of Zürich, the Swiss National Science Foundation (SNSF) (projects n°200021_132521; n°*PBZHP*2_147309), as well as by the EU FP7 People Programme (Marie Curie Actions) under REA Grant Agreement n°290227.

Furthermore, the authors would like to acknowledge the Computer-Assisted Paleoanthropology group and the Visualization and MultiMedia Lab at University of Zürich for providing the hazelnut volume; volvis.org [1] for the human foot and bonsai; Prof. Dr. Schittny from the Institute of Anatomy at University of Bern for the lung; and Anderson et al. [3] for the video.

References

[1] Real world medical datasets. http://volvis.org/.

[2] vmmlib: A Vector and Matrix Math Library. Visualization and MultiMedia Lab, University of Zurich. Available online; 2014. URL: `http://vmml.github.io/vmmlib/`.

[3] Anderson D, Luke RH, Keller JM, Skubic M, Rantz M, Aud M. Linguistic summarization of video for fall detection using voxel person and fuzzy logic. Comput Vis Image Underst 2009;113(1):80–9.

[4] Bader BW, Kolda TG. Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. ACM Transactions on Mathematical Software 2006;32(4):635–53.

[5] Bader BW, Kolda TG, et al. MATLAB tensor toolbox version 2.5. Available online; 2012. URL: `http://www.sandia.gov/~tgkolda/TensorToolbox/`.

[6] Balsa Rodríguez M, Gobbetti E, Iglesias Guitiàn JA, Makhinya M, Marton F, Pajarola R, Suter SK. State-of-the-art in compressed GPU-based direct volume rendering. Computer Graphics Forum 2014;33(6):77–100.

[7] Berge J. The typical rank of tall three-way arrays. Psychometrika 2000;65(4):525–32.

[8] Bilgili A, Öztürk A, Kurt M. A general BRDF representation based on tensor decomposition. Computer Graphics Forum 2011;30(8):2427–39.

[9] Carroll JD, Chang JJ. Analysis of individual differences in multidimensional scaling via an *n*-way generalization of "Eckart–Young" decompositions. Psychometrika 1970;35:283–319.

[10] Eckart C, Young G. The approximation of one matrix by another of lower rank. Psychometrika 1936;1(3):211–8.

[11] Engel K, Hadwiger M, Kniss JM, Rezk-Salama C, Weiskopf D. Real-Time Volume Graphics. AK Peters, 2006.

[12] Ergin S, Çakir S, Gerek ON, Gülmezoğlu MB. A new implementation of common matrix approach using third-order tensors for face recognition. Expert Systems with Applications 2011;38(4):3246–51.

[13] Fout N, Ma KL. Transform coding for hardware-accelerated volume rendering. IEEE Transaction on Visualization and Computer Graphics 2007;13(6):1600–7.

[14] Harshman RA. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi–modal factor analysis. UCLA working papers in phonetics 1970;16:1–84.

[15] He X, Cai D, Liu H, Han J. Image clustering with tensor representation. In: Proceedings ACM International Conference on Multimedia. 2005. p. 132–40.

[16] Hitchcock FL. The expression of a tensor or a polyadic as a sum of products. Journal of Mathematics and Physics 1927;6(1):164–89.

[17] Kiers HAL. Towards a standardized notation and terminology in multiway analysis. Journal of Chemometrics 2000;14(3):105–22.

[18] Kolda TG, Bader BW. Tensor decompositions and applications. SIAM Review 2009;51(3):455–500.

[19] de Lathauwer L. Decompositions of a higher-order tensor in block terms – Part II: Definitions and uniqueness. SIAM Journal on Matrix Analysis and Applications 2008;30(3):1033–66.

[20] de Lathauwer L, de Moor B, Vandewalle J. A multilinear singular value decomposition. SIAM Journal of Matrix Analysis and Applications 2000;21(4):1253–78.

[21] de Lathauwer L, de Moor B, Vandewalle J. On the best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of higher-order tensors. SIAM Journal of Matrix Analysis and Applications 2000;21(4):1324–42.

[22] de Lathauwer L, Nion D. Decompositions of a higher-order tensor in block terms – Part III: Alternating least squares algorithms. SIAM Journal of Matrix Analysis and Applications 2008;30(3):1067–83.

[23] Morozov OV, Unser M, Hunziker P. Reconstruction of large, irregularly sampled multidimensional images. A tensor-based approach. IEEE Transactions on Medical Imaging 2011;30(2):366–74.

[24] Ruiters R, Schwartz C, Klein R. Data Driven Surface Reflectance from Sparse and Irregular Samples. Computer Graphics Forum 2012;31(2):315–24.

[25] Schultz T, Seidel HP. Estimating crossing fibers: A tensor decomposition approach. IEEE Transactions on Visualization and Computer Graphics 2008;14(6):1635–42.

[26] Shashua A, Hazan T. Non-negative tensor factorization with applications to statistics and computer vision. In: Proceedings of the International Conference on Machine Learning. ICML; 2005. p. 792–9.

[27] Sidiropoulos ND, Bro R. On the uniqueness of multilinear decomposition of *N*-way arrays. Journal of Chemometrics 2000;14:229–39.

[28] Suter SK, Iglesias Guitián JA, Marton F, Agus M, Elsener A, Zollikofer CP, Gopi M, Gobbetti E, Pajarola R. Interactive multiscale tensor reconstruction for multiresolution volume visualization. IEEE Transactions on Visualization and Computer Graphics 2011;17(12):2135–43.

[29] Suter SK, Makhinya M, Pajarola R. TAMRESH: Tensor approximation multiresolution hierarchy for interactive volume visualization. Computer Graphics Forum 2013;32(8):151–60.

[30] Suter SK, Zollikofer CP, Pajarola R. Application of tensor approximation to multiscale volume feature representations. In: Proceedings Vision, Modeling and Visualization. 2010. p. 203–10.

[31] Suter SK, Zollikofer CP, Pajarola R. Multiscale Tensor Approximation for Volume Data. Technical Report IFI-2010.04; Department of Informatics, University of Zürich; 2010.

[32] Tsai YT, Fang KL, Lin WC, Shih ZC. Modeling bidirectional texture functions with multivariate spherical radial basis functions. IEEE Transactions on Pattern Analysis and Machine Intelligence 2011;33(7):1356–69.

[33] Tsai YT, Shih ZC. K-clustered tensor approximation: A sparse multilinear model for real-time rendering. ACM Transactions on Graphics 2012;31(3):1–17.

[34] Tucker LR. Implications of factor analysis of three-way matrices for measurement of change. In: Problems in measuring change. Madison WI: University of Wisconsin Press; 1963. p. 122–37.

[35] Tucker LR. Some mathematical notes on three-mode factor analysis. Psychometrika 1966;31(3):279–311.

[36] Vasilescu MAO, Terzopoulos D. Multilinear analysis of image ensembles: TensorFaces. In: Proceedings European Conference on Computer Vision. 2002. p. 447–60.

[37] Vasilescu MAO, Terzopoulos D. TensorTextures: Multilinear image-based rendering. ACM Transactions on Graphics 2004;23(3):336–42.

[38] Wang H, Ahuja N. Rank-R approximation of tensors: Using image-as-matrix representation. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition. 2005. p. 346–53.

[39] Wang H, Ahuja N. A tensor approximation approach to dimensionality reduction. International Journal of Computer Vision 2008;76(3):217–29.

[40] Wang H, Wu Q, Shi L, Yu Y, Ahuja N. Out-of-core tensor approximation of multi-dimensional matrices of visual data. ACM Transactions on Graphics 2005;24(3):527–35.

[41] Wu Q, Xia T, Chen C, Lin HYS, Wang H, Yu Y. Hierarchical tensor approximation of multidimensional visual data. IEEE Transactions on Visualization and Computer Graphics 2008;14(1):186–99.

[42] Yan S, Wang H, Tu J, Tang X, Huang TS. Mode-kn factor analysis for image ensembles. IEEE Transactions on Image Processing 2009;18(3):670–6.

[43] Zhang T, Golub GH. Rank-one approximation to high order tensors. SIAM Journal of Matrix Analysis and Applications 2001;23(2):534–50.